UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/790,663	03/01/2004	Izydor Gryko	MSFT-2767/305783.01	9551

41505 7590 06/10/2009

WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION)
CIRA CENTRE, 12TH FLOOR
2929 ARCH STREET
PHILADELPHIA, PA 19104-2891

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
06/10/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/790,663	GRYKO ET AL.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *16 April 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-5,7-12,14-19 and 21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-5, 7-12, 14-19, and 21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's amendment dated April 16, 200, responding to the Office action mailed January 16, 2009 provided in the rejection of claims 1-5, 7-12, 14-19, and 21, wherein claims 1-5, 7-12, 14-19, and 21 have been amended.

Claims 1-5, 7-12, 14-19, and 21 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are not persuasive, thus the previous rejections are maintained and reproduced below.

Please see the section of "Response to Arguments" for details.

2. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-5, 7-12, 14-19, and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Leach et al. (Pat. No. US 6,412,020 B1) (hereinafter 'Leach') in view of Atul Gupta (*Building a Custom Project Wizard in Visual Studio .NET, May 2003, Infosys Technologies Limited*) (hereinafter 'Gupta') and Dardinski et al. (Pat. No. US 6,754,885 B1) (hereinafter 'Dardinski')

4. **As to claim 1** (Currently Amended), Leach discloses a method for building an extensible project system (e.g., Abstract, Lines 1-4 – the method aggregates an enclosed object within an enclosing object; Fig. 7A; Col. 26, Lines 33-46 - … Fig. 7A is a block diagram of an instance of multiple object. The object MTO 7A01 implements an exposed interface, the *IMultitype* interface MT, and a controlling *IUnknown* … The multitype object uses these lists to invoke the various interfaces of its enclosed aggregate objects through the multitype object's controlling *IUnknown* interface …) comprising:

- providing a base software development project component object comprising data for creating a software development project system (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while

exposing an interface of the enclosed object to client of the enclosing object; Col. 9, Lines 27-30 – to provide a method and system for enclosing objects wherein an enclosed object can itself be an enclosing object to an arbitrary level of enclosing; Col.9, Lines 45-46 – implementing controlling behavior over common functionality present in enclosed objects; Col. 10, Lines 9-13 – an enclosed object is implement with knowledge of the external interfaces of the enclosed object and has no knowledge of interfaces (other than the controlling object management interface; Col. 10, Lines 35-38 – during creation, a pointer to the enclosing multi-type object is passed to the object to be enclosed to enable the enclosed object to communicate with the enclosing multi-type object) of the enclosing object or other enclosed objects);

- providing at least one flavor component object comprising data for modifying said project system for a specific purpose (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while exposing an interface of the enclosed object to a client of the enclosing object; Col. 9, Lines 13-14 – enclosing an object within another object after the enclosing object is instantiated; Col. 9, Lines 39-42 – supplying default functionality to objects by enclosing them within an enclosing object where an enclosed or enclosing object implements the default functionality; Col. 9, Lines 55-58 – the enclosed object has an object management interface and on or more external interfaces, while the enclosing object has a controlling object management interface); and

- creating a flavored project system adapted for said specific purpose by component object aggregation using said base project component object as a participating component object and one of said at least one flavor component objects as a controlling component object (e.g., Col. 8, Lines 66-67 – a method and system for aggregating objects; Col. 9, Lines 4-7 – dynamically aggregating objects; statically aggregating objects; Col. 9, Lines 25-26 – implementing an aggregate object so that a client is unaware that the object is an aggregate; Col. 9, Lines 50-61 – the method aggregates an enclosed object within an enclosing object; each interface exposed to a client by the aggregate object has a query function member for receiving an identifier of an interface and for returning a reference to the identified interface; Col. 10, Lines 8-13 (static aggregation), 17-24 (dynamic aggregation); Col. 10, Lines 24-30 – the multi-type object has an add interface function member, which can be used to aggregate interfaces by adding them to the enclosing multi-type object; the multi-type object also has an add object function member for aggregating all of the interface of an object; Col. 10, 34-47 – a preferred method invokes the add interface function member or the add object function member of the enclosing multi-type object passing it a reference to the created object implementing the interface to be aggregated; the query function member of the enclosing multi-type object is invoked in order to retrieve a reference to the interface that has been aggregated)

Further, Leach discloses that in dynamic aggregation, rules for determining to which interface to return a reference can be added to the enclosing object and used by the query function member of the controlling object management interface (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Building a Custom Project Wizard in Visual Studio .NET*, Gupta discloses the followings:

- wherein the base project component object implements a base project configuration component object that includes configuration properties for the base project component object, the configuration properties comprising at least one of a configuration set, the configuration set comprising an indication of a build of the project system, an output file to be created, or an indication as to where the output filed will be placed (e.g., P. 3, Lines 11, 20 - *OutputPath*);
- wherein the at least one flavor component object includes flavor-specific project configuration properties, wherein the at least one flavor object includes flavor-specific project configuration properties comprising at least one of a flavor set, the flavor set comprising a caption of a project node, an icon of a project node, a property allowing a project browse object to be completely overridden, a control allowing a project to be renamed, a sort priority control, a property allowing a command to be added, a property allowing a command to be removed, a property allowing a command to be disabled, a filter, a property

allowing a default generator given a file extension to be determined, and a property allowing a human-readable generator name to be mapped to a COM object (e.g., Fig. 1 – Highlight of values set in the VSDIR file when opening a new C# project; P. 2, Bullet 3 - … the display strings and the icon are loaded …; Bullet 5 - … to be registered for use with COM … enable the configuration settings for this in the project build properties …; Note - … for the *RegisterForComInterop* setting …)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Gupta into the Leach's system to further provide other limitations stated above in the Leach's system.

The motivation is that it would further enhance the Leach's system by taking, advancing and/or incorporating Gupta's system which offers significant advantages to extend Visual Studio to create a custom project type for new types of applications, or to enforce corporate standards as once suggested by Gupta (e.g., Summary)

Furthermore, Gupta discloses extending Visual Studio to create a custom project type for new types of applications, or enforcing corporation standards (e.g. Summary) but Leach and Gupta do not explicitly disclose other limitations stated below.

However, in an analogous art of *Methods and Apparatus for Controlling Object Appearance in a Process Control Configuration System*, Dardinski discloses the followings:

- signaling by the base project component object to the at least one flavor component object that the base project configuration component object needs to be extended; and
- creating, by the at least one flavor component object, a flavored base project configuration component object, wherein at least one configuration property for the base project component object is modified by a corresponding flavor-specific project configuration property (e.g., Col. 3, Lines 47-55 - the configurable objects of such an apparatus can be associated with one another in a hierarchical relationship, such that at least one such object is a descendant of another; descendants inherit parameters from their ancestors; inherited information may be overridden)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Dardinski into the Leach-Gupta's system to further provide other limitations stated above in the Leach-Gupta's system.

The motivation is that it would further enhance the Leach-Gupta's system by taking, advancing and/or incorporating Dardinski's system which offers significant advantages that the configurable objects of such an apparatus can be associated with one another in a hierarchical relationship, such that at least one

such object is a descendant of another; descendants inherit parameters from their ancestors; inherited information may be overridden as once suggested by Dardinski (e.g., Col. 3, Lines 47-55)

5. **As to claim 2** (Currently Amended) (incorporating the rejection in claim 1), Leach discloses the method where said at least one flavor component object comprises at least a first flavor component object and a second flavor component object, and where said step of creating a flavored project system comprises: creating an intermediary component object by aggregating said first flavor component object as a controlling component object and said base project component object as a participating component object; and creating a flavored project system by using said second flavor component object as a controlling component object and said intermediary component object as a participating object (e.g., Col. 9, Lines 27-30 – to provide a method and system for enclosing objects where an enclosed object can itself be an enclosing object to an arbitrary level of enclosing).

6. **As to claim 3** (Currently Amended) (incorporating the rejection in claim 1), Leach discloses the method where said step of creating a flavored project system comprises allowing at least one interface of said base project to be modified by said flavor component object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these

combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object)

7. **As to claim 4** (Currently Amended) (incorporating the rejection in claim 3), Leach discloses the method where said step of creating a flavored project system comprises allowing a value for at least one property stored in said at least one interface of said base project to be modified by a value for said at least one property stored in an interface of said flavor component object (e.g., Col. 5, Lines 10-12 – the overriding virtual function can modify the state of the object in a way that affects non-overridden functions; Col. 9, Lines 1-2 – to provide a method and system for dynamically modifying object behavior; Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object; Col. 10, Lines 17-24 – an object can be modified dynamically by allowing interface instances, as implemented by objects, to be aggregated during the execution of a client program).

8. **As to claim 5** (Currently Amended) (incorporating the rejection in claim 1), Leach discloses the method where said step of creating a flavored project system comprises allowing at least one interface of said base project to be

replaced by said flavor component object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object)

9. **As to claim 7** (Previously Presented) (incorporating the rejection in claim 1), Dardinski discloses the method wherein said flavored base project configuration component object (e.g., Col. 8, Line 57 through Col. 9, Line 2 - The *IinitFromTemplateStream* interface of an assembly object (i.e., a base project configuration object) has one method that controls the initialization of the assembly object from a passed stream; The initialization data is static configuration data along with the initialization data for assembly parameters. Other assembly data, such as ambient properties, can be passed to an assembly via a connection. The initialization data for the assembly parameters is passed in the steam and the static configuration data can be available through the class identifier of the assembly. The assembly can be customized through the parameters) includes an extender interface, said creation of a project system further comprising: providing an extender site component object associated with said extender interface (e.g., Fig. 1 – illustrating an assembly object along with its connection to external entities; Col. 4, Line 52 through Col. 5, Line 20 – an external entity connects assembly-2 to assembly-3 by retrieving the reference to

a connector of assembly-2 and requesting the connector to export the element identified by index "i1", represented by plug102a. the external entity then requests connector-3 of assembly-3 to connect assembly-2 through the connection identified by role "r1", represented by socket 103b)

10. **As to claim 8** (Currently Amended), Leach discloses a system for building an extensible project system (Abstract, Lines 1-4 – the method aggregates an enclosed object within an enclosing object; Fig. 7A; Col. 26, Lines 33-46 - … Fig. 7A is a block diagram of an instance of multiple object. The object MTO 7A01 implements an exposed interface, the *IMultitype* interface MT, and a controlling *IUnknown* … The multitype object uses these lists to invoke the various interfaces of its enclosed aggregate objects through the multitype object's controlling *IUnknown* interface …) comprising:

- a processor;
- a process configured to instantiate a base software development project component object comprising data for creating a project system (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while exposing an interface of the enclosed object to client of the enclosing object; Col. 9, Lines 27-30 – to provide a method and system for enclosing objects wherein an enclosed object can itself be an enclosing object to an arbitrary level of enclosing; Col.9, Lines 45-46 – implementing controlling behavior over common functionality present in enclosed objects; Col. 10, Lines 9-13 – an enclosed object is implement with knowledge of the

external interfaces of the enclosed object and has no knowledge of interfaces (other than the controlling object management interface; Col. 10, Lines 35-38 – during creation, a pointer to the enclosing multi-type object is passed to the object to be enclosed to enable the enclosed object to communicate with the enclosing multi-type object) of the enclosing object or other enclosed objects);

- a process configured to instantiate at least one flavor component object comprising data for modifying said project system for a specific purpose (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while exposing an interface of the enclosed object to a client of the enclosing object; Col. 9, Lines 13-14 – enclosing an object within another object after the enclosing object is instantiated; Col. 9, Lines 39-42 – supplying default functionality to objects by enclosing them within an enclosing object where an enclosed or enclosing object implements the default functionality; Col. 9, Lines 55-58 – the enclosed object has an object management interface and on or more external interfaces, while the enclosing object has a controlling object management interface); and
- a process configured to generate a flavored software development project system for said specific purpose by component object aggregation using said base project component object as a participating component object and one of said at least one flavor component objects as a controlling component object (e.g., Col. 8, Lines 66-67 – a method and system for aggregating objects; Col. 9, Lines 4-7 – dynamically aggregating objects;

statically aggregating objects; Col. 9, Lines 25-26 – implementing an aggregate object so that a client is unaware that the object is an aggregate; Col. 9, Lines 50-61 – the method aggregates an enclosed object within an enclosing object; each interface exposed to a client by the aggregate object has a query function member for receiving an identifier of an interface and for returning a reference to the identified interface; Col. 10, Lines 8-13 (static aggregation), 17-24 (dynamic aggregation); Col. 10, Lines 24-30 – the multi-type object has an add interface function member, which can be used to aggregate interfaces by adding them to the enclosing multi-type object; the multi-type object also has an add object function member for aggregating all of the interface of an object; Col. 10, 34-47 – a preferred method invokes the add interface function member or the add object function member of the enclosing multi-type object passing it a reference to the created object implementing the interface to be aggregated; the query function member of the enclosing multi-type object is invoked in order to retrieve a reference to the interface that has been aggregated)

Further, Leach discloses that in dynamic aggregation, rules for determining to which interface to return a reference can be added to the enclosing object and used by the query function member of the controlling object management interface (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Building a Custom Project Wizard in Visual Studio .NET*, Gupta discloses the following:

- wherein the base project component object implements a base project configuration component object that includes configuration properties for the base project component object, the configuration properties comprising at least one of a configuration set, the configuration set comprising an indication of a build of the project system, an output file to be created, or an indication as to where the output filed will be placed (e.g., P. 3, Lines 11, 20 - *OutputPath*)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Gupta into the Leach's system to further provide other limitations stated above in the Leach's system.

The motivation is that it would further enhance the Leach's system by taking, advancing and/or incorporating Gupta's system which offers significant advantages to extend Visual Studio to create a custom project type for new types of applications, or to enforce corporate standards as once suggested by Gupta (e.g., Summary)

Furthermore, Gupta discloses creating custom project template wizards in Visual Studio .NET but Leach and Gupta do not explicitly disclose other limitations stated below.

However, in an analogous art of *Methods and Apparatus for Controlling Object Appearance in a Process Control Configuration System*, Dardinski discloses the followings:

- wherein the at least one flavor component object includes flavor-specific project configuration properties (e.g., Col. 12, Lines 7-17 - ... The Parameter Values in the Modifier Parameterized Objects are used to override the parameters of the Parameterized Object ...; Col. 14, Lines 15-37 - Parameter Override; Col. 39, Lines 19 through Col. 40, Line 56 – COM Architecture in IDA – this is a powerful tool for easily building and maintaining IDA functionality, as well as giving users an extremely rich and flexible way to customize and extend IDA; Col. 64, Lines 21-24 - When user-level security is enabled ....; Col. 22, Lines 2-5 - ... Some of the attribute and assignable selections may be disable when the object ...; Col. 3, Lines 19-27 - ... Appearance objects provide icons or representations indicating how the configurable objects are to be depicted ...The placeholder objects identify the sizes, locations, colors, etc., of the icons ... to represent the configurable objects);
- a process in the base project component object configured to signal the at least one flavor component object that the base project configuration component object needs to be extended; and
- a process in the at least one flavor component object configured to generate a flavored base software development project configuration

component object, wherein at least one configuration property for the base project component object is modified by a corresponding flavor-specific project configuration property (e.g., Col. 3, Lines 47-55 - the configurable objects of such an apparatus can be associated with one another in a hierarchical relationship, such that at least one such object is a descendant of another; descendants inherit parameters from their ancestors; inherited information may be overridden)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Dardinski into the Leach-Gupta's system to further provide other limitations stated above in the Leach-Gupta's system.

The motivation is that it would further enhance the Leach-Gupta's system by taking, advancing and/or incorporating Dardinski's system which offers significant advantages that the configurable objects of such an apparatus can be associated with one another in a hierarchical relationship, such that at least one such object is a descendant of another; descendants inherit parameters from their ancestors; inherited information may be overridden as once suggested by Dardinski (e.g., Col. 3, Lines 47-55)

11. **As to claim 9** (Currently Amended) (incorporating the rejection in claim 8), Leach discloses the system where said at least one flavor component object comprises at least a first flavor component object and a second flavor component object, and where said aggregator further comprises: a first aggregator for

creating an intermediary component object by aggregating said first flavor component object as a controlling component object and said base project object as a participating object; and a second aggregator for creating a flavored project system by using said second flavor component object as a controlling component object and said intermediary component object as a participating component object (e.g., Col. 9, Lines 27-30 – to provide a method and system for enclosing objects where an enclosed object can itself be an enclosing object to an arbitrary level of enclosing)

12. **As to claim 10** (Currently Amended) (incorporating the rejection in claim 8), Leach discloses the system where said aggregator causes at least one interface of said base project to be modified by said flavor component object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object)

13. **As to claim 11** (Currently Amended) (incorporating the rejection in claim 10), Leach discloses the system where said aggregator causes a value for at least one property stored in said at least one interface of said base project to be modified by a value for said at least one property stored in an interface of said flavor component object (e.g., Col. 5, Lines 10-12 – the overriding virtual function

can modify the state of the object in a way that affects non-overridden functions; Col. 9, Lines 1-2 – to provide a method and system for dynamically modifying object behavior; Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object; Col. 10, Lines 17-24 – an object can be modified dynamically by allowing interface instances, as implemented by objects, to be aggregated during the execution of a client program)

14. **As to claim 12** (Currently Amended) (incorporating the rejection in claim 8), Leach discloses the system where said aggregator causes at least one interface of said base project to be replaced by said flavor component object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object)

15. **As to claim 14** (Currently Amended) (incorporating the rejection in claim 8), Dardinski discloses the system wherein said flavored base project configuration component object (e.g., Col. 8, Line 57 through Col. 9, Line 2 - The

IinitFromTemplateStream interface of an assembly object (i.e., a base project configuration object) has one method that controls the initialization of the assembly object from a passed stream; The initialization data is static configuration data along with the initialization data for assembly parameters. Other assembly data, such as ambient properties, can be passed to an assembly via a connection. The initialization data for the assembly parameters is passed in the steam and the static configuration data can be available through the class identifier of the assembly. The assembly can be customized through the parameters) includes an extender interface, said project system further comprising: an extender site component object associated with said extender interface (e.g., Fig. 1 – illustrating an assembly object along with its connection to external entities; Col. 4, Line 52 through Col. 5, Line 20 – an external entity connects assembly-2 to assembly-3 by retrieving the reference to a connector of assembly-2 and requesting the connector to export the element identified by index "i1", represented by plug102a. the external entity then requests connector-3 of assembly-3 to connect assembly-2 through the connection identified by role"r1", represented by socket 103b)

16. **As to claim 15** (Currently Amended), Leach discloses a computer-readable medium storage for building an extensible project system (e.g., Abstract, Lines 1-4 – the method aggregates an enclosed object within an enclosing object; Fig. 7A; Col. 26, Lines 33-46 - … Fig. 7A is a block diagram of an instance of multiple object. The object MTO 7A01 implements an exposed

interface, the *IMultitype* interface MT, and a controlling *IUnknown* ... The multitype object uses these lists to invoke the various interfaces of its enclosed aggregate objects through the multitype object's controlling *IUnknown* interface ...), said computer readable storage medium storing instructions for causing a computer to perform the steps of comprising:

- providing a base software development project component object comprising data for creating a project system (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while exposing an interface of the enclosed object to client of the enclosing object; Col. 9, Lines 27-30 – to provide a method and system for enclosing objects wherein an enclosed object can itself be an enclosing object to an arbitrary level of enclosing; Col.9, Lines 45-46 – implementing controlling behavior over common functionality present in enclosed objects; Col. 10, Lines 9-13 – an enclosed object is implement with knowledge of the external interfaces of the enclosed object and has no knowledge of interfaces (other than the controlling object management interface; Col. 10, Lines 35-38 – during creation, a pointer to the enclosing multi-type object is passed to the object to be enclosed to enable the enclosed object to communicate with the enclosing multi-type object) of the enclosing object or other enclosed objects);
- providing at least one flavor component object comprising data for modifying said project system for a specific purpose (e.g., Col. 9, Lines 9-11 – enclosing an object within another object while exposing an interface

of the enclosed object to a client of the enclosing object; Col. 9, Lines 13-14 – enclosing an object within another object after the enclosing object is instantiated; Col. 9, Lines 39-42 – supplying default functionality to objects by enclosing them within an enclosing object where an enclosed or enclosing object implements the default functionality; Col. 9, Lines 55-58 – the enclosed object has an object management interface and on or more external interfaces, while the enclosing object has a controlling object management interface); and

- creating a flavored software development project system for said specific purpose by component object aggregation using said base project component object as a participating component object and one of said at least one flavor component objects as a controlling component object (Col. 8, Lines 66-67 – a method and system for aggregating objects; Col. 9, Lines 4-7 – dynamically aggregating objects; statically aggregating objects; Col. 9, Lines 25-26 – implementing an aggregate object so that a client is unaware that the object is an aggregate; Col. 9, Lines 50-61 – the method aggregates an enclosed object within an enclosing object; each interface exposed to a client by the aggregate object has a query function member for receiving an identifier of an interface and for returning a reference to the identified interface; Col. 10, Lines 8-13 (static aggregation), 17-24 (dynamic aggregation); Col. 10, Lines 24-30 – the multi-type object has an add interface function member, which can be used to aggregate interfaces by adding them to the enclosing multi-type

object; the multi-type object also has an add object function member for aggregating all of the interface of an object; Col. 10, 34-47 – a preferred method invokes the add interface function member or the add object function member of the enclosing multi-type object passing it a reference to the created object implementing the interface to be aggregated; the query function member of the enclosing multi-type object is invoked in order to retrieve a reference to the interface that has been aggregated).

Further, Leach discloses that in dynamic aggregation, rules for determining to which interface to return a reference can be added to the enclosing object and used by the query function member of the controlling object management interface (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Building a Custom Project Wizard in Visual Studio .NET*, Gupta discloses the following:

- wherein the base project component object implements a base software development project configuration component object that includes configuration properties for the base project component object, the configuration properties comprising at least one of a configuration set, the configuration set comprising an indication of a build of the project system, an output file to be created, and an indication as to where the output file will be placed (e.g., P. 3, Lines 11, 20 - *OutputPath*)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Gupta into the Leach's system to further provide other limitations stated above in the Leach's system.

The motivation is that it would further enhance the Leach's system by taking, advancing and/or incorporating Gupta's system which offers significant advantages to extend Visual Studio to create a custom project type for new types of applications, or to enforce corporate standards as once suggested by Gupta (e.g., Summary)

Furthermore, Gupta discloses extending Visual Studio to create a custom project type for new types of applications, or enforcing corporation standards (e.g. Summary) but Leach and Gupta do not explicitly disclose other limitations stated below.

However, in an analogous art of *Methods and Apparatus for Controlling Object Appearance in a Process Control Configuration System*, Dardinski discloses the followings:

- wherein the at least one flavor component object includes flavor-specific project configuration properties (e.g., Col. 12, Lines 7-17 - … The Parameter Values in the Modifier Parameterized Objects are used to override the parameters of the Parameterized Object …; Col. 14, Lines 15-37 - Parameter Override; Col. 39, Lines 19 through Col. 40, Line 56 – COM Architecture in IDA – this is a powerful tool for easily building and maintaining IDA functionality, as well as giving users an

extremely rich and flexible way to customize and extend IDA; Col. 64, Lines 21-24 - When user-level security is enabled ....; Col. 22, Lines 2-5 - ... Some of the attribute and assignable selections may be disable when the object ...; Col. 3, Lines 19-27 - ... Appearance objects provide icons or representations indicating how the configurable objects are to be depicted ...The placeholder objects identify the sizes, locations, colors, etc., of the icons ... to represent the configurable objects);

- signaling by the base project component object to the at least one flavor component object that the base project configuration component object needs to be extended; and
- creating, by the at least one flavor component object, a flavored base software development project configuration component object, wherein at least one configuration property for the base project component object is modified by a corresponding flavor-specific project configuration property (e.g., Col. 3, Lines 47-55 - the configurable objects of such an apparatus can be associated with one another in a hierarchical relationship, such that at least one such object is a descendant of another; descendants inherit parameters from their ancestors; inherited information may be overridden)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Dardinski into the

Leach-Gupta's system to further provide other limitations stated above in the Leach-Gupta's system.

The motivation is that it would further enhance the Leach-Gupta's system by taking, advancing and/or incorporating Dardinski's system which offers significant advantages that the configurable objects of such an apparatus can be associated with one another in a hierarchical relationship, such that at least one such object is a descendant of another; descendants inherit parameters from their ancestors; inherited information may be overridden as once suggested by Dardinski (e.g., Col. 3, Lines 47-55)

17. **As to claim 16** (Currently Amended) (incorporating the rejection in claim 15), Leach discloses the computer-readable medium storage where said at least one flavor component object comprises at least a first flavor component object and a second flavor component object, and where said step of creating a flavored project system comprises: creating an intermediary component object by aggregating said first flavor component object as a controlling component object and said base project component object as a participating object; and creating a flavored project system by using said second flavor component object as a controlling component object and said intermediary component object as a participating component object (e.g., Col. 9, Lines 27-30 – to provide a method and system for enclosing objects where an enclosed object can itself be an enclosing object to an arbitrary level of enclosing)

18. **As to claim 17** (Currently Amended) (incorporating the rejection in claim 15), Leach discloses the computer-readable medium storage where said step of creating a flavored project system comprises allowing at least one interface of said base project to be modified by said flavor component object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object)

19. **As to claim 18** (Currently Amended) (incorporating the rejection in claim 17), Leach discloses the computer-readable medium storage where said step of creating a flavored project system comprises allowing a value for at least one property stored in said at least one interface of said base project to be modified by a value for said at least one property stored in an interface of said flavor component object (e.g., Col. 5, Lines 10-12 – the overriding virtual function can modify the state of the object in a way that affects non-overridden functions; Col. 9, Lines 1-2 – to provide a method and system for dynamically modifying object behavior; Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object; Col.

10, Lines 17-24 – an object can be modified dynamically by allowing interface instances, as implemented by objects, to be aggregated during the execution of a client program)

20. **As to claim 19** (Currently Amended) (incorporating the rejection in claim 15), Leach discloses the computer-readable medium storage where said step of creating a flavored project system comprises allowing at least one interface of said base project to be replaced by said flavor component object (e.g., Col. 9, Lines 34-37 – for enhancing a base object's apparent behavior by adding an interface to it that overrides standard behavior of the base object; Col. 25, Lines 3-16 – these combining rules can be used to override the standard behavior of an enclosed base object by providing access to a new implementation of a previously defined interface of the enclosed base object)

21. **As to claim 21** (Currently Amended) (incorporating the rejection in claim 15), Dardinski discloses the computer-readable storage medium wherein said flavored base project configuration component object (e.g., Col. 8, Line 57 through Col. 9, Line 2 - The *IinitFromTemplateStream* interface of an assembly object (i.e., a base project configuration object) has one method that controls the initialization of the assembly object from a passed stream; The initialization data is static configuration data along with the initialization data for assembly parameters. Other assembly data, such as ambient properties, can be passed to an assembly via a connection. The initialization data for the assembly

parameters is passed in the steam and the static configuration data can be available through the class identifier of the assembly. The assembly can be customized through the parameters) includes an extender interface, said creation of a project system further comprising: providing an extender site component object associated with said extender interface (e.g., Fig. 1 – illustrating an assembly object along with its connection to external entities; Col. 4, Line 52 through Col. 5, Line 20 – an external entity connects assembly-2 to assembly-3 by retrieving the reference to a connector of assembly-2 and requesting the connector to export the element identified by index "i1", represented by plug102a. the external entity then requests connector-3 of assembly-3 to connect assembly-2 through the connection identified by role "r1", represented by socket 103b)

Response to Arguments

22. Applicant's arguments filed on April 16, 2009 have been fully considered but they are not persuasive.

In the remarks, Applicant argues that, for examples:

(A.1) Applicants have amended independent claims 1, 8 and 15 to clarify that it claims the use of 'component object ' or 'component'; However, *Leach* appears to use the term 'object' to refer exclusively to an object in an object oriented programming language (emphasis added)

Examiner's response:

(R.1) Firstly, the supported paragraphs [0033] and [0034] recited in last full paragraph of page 8 in the REMARKS for 'component object' are corresponding to the PGPUB document (US 2005/0193383). In the specification, they are paragraphs [0029] and [0030] instead.

Secondly, the paragraph [0033], based on the PGPUB, recites that "... Object models, such as Microsoft Corporation's Component Object Model ('COM'), facilitate the implementation of complex systems by allowing the use of objects as reusable components. COM is more fully described in 'Inside COM' by Dale Rogerson and published by Microsoft Press in 1997. COM specifies that each object is to implement a certain interface referred to as the *IUnknown* interface. The IUnknown interface in COM provides a query interface function, an add reference function, and a release function ..." (emphasis added)

Thirdly, *Leach* indeed uses Component Object Model (COM). For example, the Exhibit A below, excerpted from Figure 7A in the Leach's reference, clearly shows the uses of '*IUnknown*' which is required for 'COM' architecture (emphasis added). Additionally, *Leach* teaches "... Fig. 7A is a block diagram of an instance of multiple object. The object MTO 7A01 implements an exposed interface, the *IMultitype* interface MT, and a controlling *IUnknown* ... The multitype object uses these lists to invoke the various interfaces of its enclosed aggregate objects through the multitype object's controlling *IUnknown* interface ..." (recited in Column 26, Lines 33-46 – emphasis added)

Thus, *Leach* teaches the use of 'component object'.

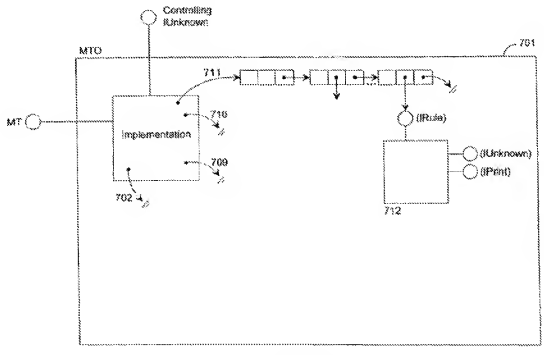

Fig. 7A

Exhibit A

Conclusion

23. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Ben C. Wang
Examiner, Art Unit 2192

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192